



NORTH-HOLLAND

# Binary Join Trees for Computing Marginals in the Shenoy-Shafer Architecture

**Prakash P. Shenoy**

*School of Business, University of Kansas, Lawrence, Kansas*

---

## ABSTRACT

*We describe a data structure called binary join trees that is useful in computing multiple marginals efficiently in the Shenoy-Shafer architecture. We define binary join trees, describe their utility, and describe a procedure for constructing them. © 1997 Elsevier Science Inc.*

**KEYWORDS:** *local computation, fusion algorithm, join trees, binary join trees*

---

## 1. INTRODUCTION

The main goal of this paper is to describe a data structure called binary join trees that is useful in computing multiple marginals efficiently in the Shenoy-Shafer architecture [31]. We define binary join trees, describe their utility, and describe a procedure for constructing them.

In the last decade, much work has been done in the uncertain-reasoning community on exact computation of marginals using local computation (see, e.g., [18, 11, 12, 8, 31, 9, 1, 13, 21]). The main idea behind local computation is to compute marginals of the joint distribution without actually computing the joint distribution. Local computation can be described as message passing in data structures called join trees. Join trees are also called junction trees [9], clique trees [12], qualitative Markov trees [22], and hypertrees [31].

---

*Address correspondence to Professor Prakash P. Shenoy, School of Business, University of Kansas, Summerfield Hall, Lawrence, KS 66045-2003. E-mail: pshenoy@ukans.edu.*

Received August 1, 1996; accepted October 1, 1996.

The efficiency of the message-passing algorithms depend on the sizes of the subsets in a join tree. The problem of finding a join tree that minimizes the size of the largest subset has been shown to be NP-complete [2]. Consequently, much attention has been devoted to finding heuristics for constructing good join trees (see, e.g., [17, 11, 16, 34, 10]).

In this paper, we focus on another aspect of join trees, the number of neighbors of nodes in a join tree. If a node in a join tree has many neighbors, then it leads to much inefficiency, especially in the Shenoy-Shafer architecture. This motivates the definition of the binary join tree which is a join tree such that no node has more than three neighbors. The main idea behind a binary join tree is that all combinations are done on a binary basis, i.e., we combine functions two at a time.

Local computation has also been studied in many other domains besides uncertain reasoning, such as solving systems of equations [19], optimization [4], and relational databases [3]. In order to keep the applicability of the results as wide as possible, we describe our work using the abstract framework of valuation networks [23, 26, 28–30].

An outline of this paper is as follows. Section 2 introduces the framework of valuation networks. Section 3 describes three axioms that enable local computation. Section 4 describes a fusion algorithm for computing a marginal of the joint valuation. Section 5 describes the fusion algorithm in terms of message passing in join trees. Section 6 describes a message-passing algorithm for computing multiple marginals—the Shenoy-Shafer architecture. Section 7 introduces the concept of binary join trees, its utility in reducing the number of combinations done in computing marginals, and a procedure for constructing them. Finally, Section 8 contains concluding remarks.

---

## 2. THE VALUATION NETWORK FRAMEWORK

---

This section describes the abstract valuation network (VN) framework. In a VN, we represent knowledge by entities called valuations, and we make inferences using two operators called marginalization and combination that operate on valuations.

### Variables and Configurations

We use the symbol  $\Omega_X$  for the set of possible values of a variable  $X$ , and we call  $\Omega_X$  the *state space* for  $X$ . We are concerned with a finite set  $\Psi$  of variables, and we assume that all the variables in  $\Psi$  have finite state spaces. We use uppercase Latin letters such as  $X, Y, Z$ , etc., to denote variables, and we use lowercase Latin letters such as  $r, s, t$ , etc., to denote sets of variables.

Given a nonempty set  $s$  of variables, let  $\Omega_s$  denote the Cartesian product of  $\Omega_X$  for  $X \in s$ ;  $\Omega_s = \times \{\Omega_X \mid X \in s\}$ . We call  $\Omega_s$  the *state space* for  $s$ . We call the elements of  $\Omega_s$  *configurations* of  $s$ . We use lowercase boldface letters such as  $\mathbf{x}, \mathbf{y}, \mathbf{z}$ , etc., to denote configurations.

It is convenient to extend this terminology to the case where the set  $s$  is empty. We adopt the convention that the state space for the empty set  $\emptyset$  consists of a single configuration, and we use the symbol  $\blacklozenge$  to name that configuration:  $\Omega_\emptyset = \{\blacklozenge\}$ .

### Valuations

Given a subset  $s$  of variables (possibly empty), there is a set  $\vartheta_s$ . We call the elements of  $\vartheta_s$  *valuations for  $s$* . Let  $\vartheta$  denote the set of all valuations, i.e.,  $\vartheta = \cup \{\vartheta_s \mid s \subseteq \Psi\}$ . If  $\sigma$  is a valuation for  $s$ , we say  $s$  is the *domain* of  $\sigma$ . We use lowercase Greek letters such as  $\rho, \sigma, \tau$ , etc., to denote valuations.

Valuations are primitives in the VN framework and as such require no definition. But, as we shall see shortly, they are entities that can be combined (with other valuations) and marginalized. Intuitively, a valuation for  $s$  represents some knowledge about the variables in  $s$ .

In probability theory, valuations are called probability potentials. A *probability potential* for  $r$  is a function  $\rho: \Omega_r \rightarrow [0, 1]$ . In the Dempster-Shafer belief-function theory, valuations are called bpa potentials. A *bpa potential* for  $m$  is a function  $\mu: 2^{\Omega_m} \rightarrow [0, 1]$ , where  $2^{\Omega_m}$  denotes the set of all nonempty subsets of  $\Omega_m$ . In Spohn's epistemic belief theory, valuations are called disbelief potentials. A *disbelief potential* for  $d$  is a function  $\delta: \Omega_d \rightarrow \mathbf{N}$ , where  $\mathbf{N}$  is the set of natural numbers. In Zadeh's possibility theory, valuations are called possibility potentials. A *possibility potential* for  $p$  is a function  $\pi: \Omega_p \rightarrow [0, 1]$ .

### Marginalization

We assume that for each nonempty  $s \subseteq \Psi$ , and for each  $X \in s$ , there is a mapping  $\downarrow(s - \{X\}): \vartheta_s \rightarrow \vartheta_{s - \{X\}}$ , called *marginalization* to  $s - \{X\}$ , such that if  $\sigma$  is a valuation for  $s$ , then  $\sigma^{\downarrow(s - \{X\})}$  is a valuation for  $s - \{X\}$ . We call  $\sigma^{\downarrow(s - \{X\})}$  the *marginal* of  $\sigma$  for  $s - \{X\}$ .

Intuitively, marginalization corresponds to coarsening of knowledge. If  $\sigma$  is a valuation for  $s$  representing some knowledge about variables in  $s$ , and  $X \in s$ , then  $\sigma^{\downarrow(s - \{X\})}$  represents the knowledge about variables in  $s - \{X\}$  implied by  $\sigma$  if we disregard the variable  $X$ .

For probability potentials, marginalization from  $s$  to  $s - \{X\}$  is addition over the state space for  $X$ . For bpa potentials, marginalization from  $s$  to  $s - \{X\}$  is addition over the subsets of the state space for  $s$  that include  $\{X\}$ . For disbelief potentials, marginalization from  $s$  to  $s - \{X\}$  is mini-

mization over the state space for  $X$ . And for possibility potentials, marginalization from  $s$  to  $s - \{X\}$  is maximization over the state space for  $X$ .

### Combination

We assume there is a mapping  $\otimes : \vartheta \times \vartheta \rightarrow \vartheta$ , called *combination*, such that if  $\rho$  and  $\sigma$  are valuations for  $r$  and  $s$ , respectively, then  $\rho \otimes \sigma$  is a valuation for  $r \cup s$ .

Intuitively, combination corresponds to aggregation of knowledge. If  $\rho$  and  $\sigma$  are valuations for  $r$  and  $s$  representing knowledge about variables in  $r$  and  $s$ , respectively, then  $\rho \otimes \sigma$  represents the aggregated knowledge about variables in  $r \cup s$ .

For probability potentials, combination is pointwise multiplication followed by normalization if normalization is possible, and no normalization if normalization is not possible. For bpa potentials, combination is the product-intersection rule followed by normalization if normalization is possible, and no normalization if normalization is not possible. This rule is also known as *Dempster's rule* [7]. For disbelief potentials, combination is pointwise addition followed by normalization [24]. And for possibility potentials, combination is multiplication followed by normalization if normalization is possible, and no normalization if normalization is not possible [22].

In summary, a *valuation network* consists of a 5-tuple  $\{\Psi, \{\Omega_X\}_{X \in \Psi}, \{\tau_1, \dots, \tau_m\}, \downarrow, \otimes\}$  where  $\Psi$  is a set of variables,  $\{\Omega_X\}_{X \in \Psi}$  is a collection of state spaces,  $\{\tau_1, \dots, \tau_m\}$  is a collection of valuations,  $\downarrow$  is the marginalization operator, and  $\otimes$  is the combination operator.

**EXAMPLE 1** (Chest clinic) Consider Lauritzen and Spiegelhalter's [12] medical example:

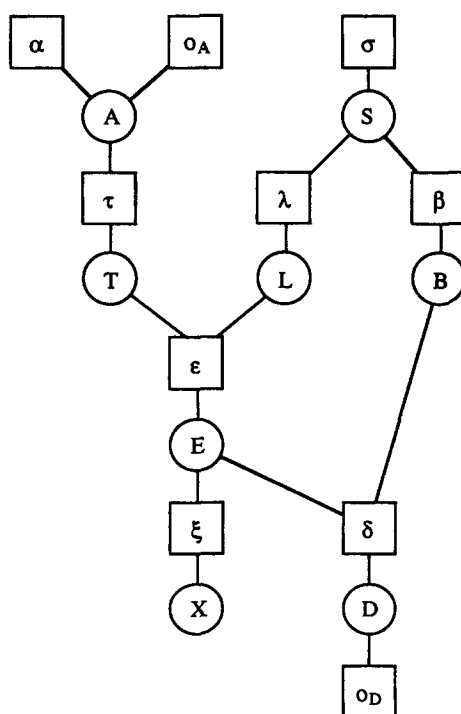
Shortness-of-breath (dyspnoea) may be due to tuberculosis, lung cancer or bronchitis, or none of them, or more than one of them. A recent visit to Asia increases the chances of tuberculosis, while smoking is known to be a risk factor for both lung cancer and bronchitis. The results of a single chest X-ray do not discriminate between lung cancer and tuberculosis, neither does the presence or absence of dyspnoea.

This needs to be applied to the following hypothetical situation. A patient presents at a chest clinic with dyspnoea, and has recently visited Asia. The doctor would like to know the chance that each of the diseases is present.

In this example, we have eight variables— $A$  (visit to Asia),  $S$  (smoking),  $T$  (tuberculosis),  $L$  (lung cancer),  $B$  (bronchitis),  $E$  (either tuberculosis or lung cancer),  $X$  (positive X-ray), and  $D$  (dyspnoea). Prior to any observations, we have eight valuations— $\alpha$  for  $\{A\}$ ,  $\sigma$  for  $\{S\}$ ,  $\tau$  for  $\{A, T\}$ ,  $\lambda$  for  $\{S, L\}$ ,  $\beta$  for  $\{S, B\}$ ,  $\varepsilon$  for  $\{T, L, E\}$ ,  $\xi$  for  $\{E, X\}$ , and  $\delta$  for  $\{E, B, D\}$ . In VNs, observations are also modeled as valuations. Suppose, for example, we observe that a patient has visited Asia recently, and is suffering from dyspnoea. These two observations can be modeled as valuations  $o_A$  for  $\{A\}$  and  $o_D$  for  $\{D\}$ , respectively.

### Valuation Networks

A graphical display of a valuation network is also called a valuation network. In such a graph, variables are represented by circular nodes, and valuations are represented by rectangular nodes. Also, each valuation node is connected by an undirected edge to each variable node in its domain. Figure 1 shows the valuation network for Example 1.



**Figure 1.** The valuation network for Example 1.

### Inferences in a VN

In a VN, the combination of all valuations is called the *joint valuation*. Given a VN, we make inferences by computing the marginal of the joint valuation for each variable of interest.

If there are  $n$  variables in a VN, and each variable has two configurations in its state space, then there are  $2^n$  configurations of all variables. Hence, it is not computationally tractable to compute the joint valuation when there are a large number of variables. In Section 4, we describe an algorithm for computing the marginal of the joint valuation for a variable without explicitly computing the joint valuation. To ensure that this algorithm gives us the correct answers, we require that marginalization and combination satisfy some axioms. The axioms are described in the next section.

---

### 3. AXIOMS FOR LOCAL COMPUTATION

---

In this section, we state three axioms that enable efficient local computation of marginals of the joint valuation. These axioms were first formulated by Shenoy and Shafer [31]. Other axiomatic systems have been defined by Shafer [20] and Cano et al. [6].

**AXIOM A1** (Order of deletion does not matter) *Suppose  $\sigma$  is a valuation for  $s$ , and suppose  $X_1, X_2 \in s$ . Then*

$$(\sigma \downarrow (s - \{X_1\})) \downarrow (s - \{X_1, X_2\}) = (\sigma \downarrow (s - \{X_2\})) \downarrow (s - \{X_1, X_2\}).$$

**AXIOM A2** (Commutativity and associativity of combination) *Suppose  $\rho, \sigma$ , and  $\tau$  are valuations for  $r, s$ , and  $t$ , respectively. Then*

$$\rho \otimes \sigma = \sigma \otimes \rho \quad \text{and} \quad \rho \otimes (\sigma \otimes \tau) = (\rho \otimes \sigma) \otimes \tau,$$

**AXIOM A3** (Distributivity of marginalization over combination) *Suppose  $\rho$  and  $\sigma$  are valuations for  $r$  and  $s$ , respectively, suppose  $X \in s$ , and suppose  $X \notin r$ . Then*

$$(\rho \otimes \sigma) \downarrow ((r \cup s) - \{X\}) = \rho \otimes (\sigma \downarrow (s - \{X\})).$$

If we regard marginalization as a coarsening of a valuation by deleting variables, then Axiom A1 says that the order in which the variables are deleted does not matter. Thus variables can be deleted in any order. One implication of this axiom is that  $(\sigma \downarrow (s - \{X_1\})) \downarrow (s - \{X_1, X_2\})$  can be written simply as  $\sigma \downarrow (s - \{X_1, X_2\})$ , i.e., we need not indicate the order in which the variables are deleted.

Axiom A2 allows us to combine valuations in any order. One implication of this axiom is that when we have multiple combinations of valuations, we can write the result without using parenthesis. For example,  $(\cdots((\sigma_1 \otimes \sigma_2) \otimes \sigma_3) \otimes \cdots \otimes \sigma_m)$  can be written simply as  $\otimes\{\sigma_i | i = 1, \dots, m\}$  or as  $\sigma_1 \otimes \cdots \otimes \sigma_m$ , i.e., we need not indicate the order in which the combinations are carried out. Another implication of Axiom A2 is that the set of valuations and the combination operator can be regarded as a commutative semigroup.

Axiom A3 is the axiom that makes local computation possible. Axiom A3 states that computation of  $(\rho \otimes \sigma)^{\downarrow((r \cup s) - \{X\})}$  can be accomplished without having to compute  $\rho \otimes \sigma$ . Notice that  $\rho \otimes \sigma$  is a valuation for  $r \cup s$  whereas  $\rho \otimes (\sigma^{\downarrow(s - \{X\})})$  is a valuation for  $(r \cup s) - \{X\}$ .

---

#### 4. COMPUTING A MARGINAL USING THE FUSION ALGORITHM

---

In this section, we describe the fusion algorithm for computing the marginal for a variable using local computation [26]. The fusion algorithm was first described by Cannings et al. [5] in the context of probability models in genetics, and they called their procedure “peeling.”

Suppose  $\{\tau_1, \dots, \tau_m\}, \downarrow, \otimes$  is a VN with  $m$  valuations. Suppose that marginalization and combination satisfy the three axioms stated in Section 4. Suppose we need to compute the marginal of the joint valuation for subset  $t$ ,  $(\tau_1 \otimes \cdots \otimes \tau_m)^{\downarrow t}$ . The basic idea of the fusion algorithm is to successively delete all variables in  $\Psi - t$  from the VN. The variables may be deleted in any sequence. Axiom A1 tells us that all deletion sequences lead to the same answer. But different deletion sequences may involve different computational efforts. We will comment on good deletion sequences at the end of this section.

First let us deal with the case of deleting one variable. Suppose we have a set of  $k$  valuations  $\sigma_1, \dots, \sigma_k$ . Suppose  $\sigma_i$  is a valuation for  $s_i$ . Then  $\sigma_1 \otimes \cdots \otimes \sigma_k$  is a valuation for  $s_1 \cup \cdots \cup s_k$ . Let  $Y \in s_1 \cup \cdots \cup s_k$ , and suppose we wish to delete  $Y$  from  $\sigma_1 \otimes \cdots \otimes \sigma_k$ . Lemma 1 tells us that we can accomplish this using local computation.

**LEMMA 1** *Under the assumptions of the previous paragraph,*

$$(\sigma_1 \otimes \cdots \otimes \sigma_k)^{\downarrow(s_1 \cup \cdots \cup s_k - \{Y\})} = \sigma^{\downarrow(s - \{Y\})} \otimes (\otimes\{\sigma_i | Y \notin s_i\})$$

where  $\sigma = \otimes\{\sigma_i | Y \in s_i\}$  and  $s = \cup\{s_i | Y \in s_i\}$ .

Lemma 1 follows directly from Axiom A3 by letting  $\rho = \otimes\{\sigma_i | Y \notin s_i\}$  and  $\sigma = \otimes\{\sigma_i | Y \in s_i\}$ .

If  $\sigma = \sigma_1 \otimes \cdots \otimes \sigma_k$ , then we say  $\sigma_1, \dots, \sigma_k$  are the *factors* of  $\sigma$ . If we compare the factors of  $(\sigma_1 \otimes \cdots \otimes \sigma_k)^{\downarrow(s_1 \cup \cdots \cup s_k) - \{Y\}}$  with the factors of  $\sigma_1 \otimes \cdots \otimes \sigma_k$ , we observe that in deleting  $Y$ , the factors that do not contain  $Y$  in their domains remain unchanged and the factors that contain  $Y$  in their domains are first combined and then  $Y$  is deleted from the combination. We call this operation *fusion*. A formal definition is as follows. Consider a set of  $k$  valuations  $\sigma_1, \dots, \sigma_k$ . Suppose  $\sigma_i$  is a valuation for  $s_i$ . Let  $\text{Fus}_Y\{\sigma_1, \dots, \sigma_k\}$  denote the set of valuations after fusing the valuations in the set  $\{\sigma_1, \dots, \sigma_k\}$  with respect to variable  $Y$ . Then

$$\text{Fus}_Y\{\sigma_1, \dots, \sigma_k\} = \{\sigma^{\downarrow(s - \{Y\})}\} \cup \{\sigma_i | Y \notin s_i\}$$

where  $\sigma = \otimes \{\sigma_i | Y \in s_i\}$ , and  $s = \cup \{s_i | Y \in s_i\}$ . After fusion, the set of valuations is changed as follows. All valuations that have  $Y$  in their domains are combined, and the resulting valuation is marginalized so that  $Y$  is eliminated from its domain. The valuations that do not have  $Y$  in their domains remain unchanged.

Using the definition of fusion, we can express the result of Lemma 1 as follows:

$$(\sigma_1 \otimes \cdots \otimes \sigma_k)^{\downarrow(s_1 \cup \cdots \cup s_k - \{Y\})} = \otimes \text{Fus}_Y\{\sigma_1, \dots, \sigma_k\}$$

By successively deleting variables, we can find the marginal of the joint for any subset. Axiom A1 tells us that we can use any sequence. This result is stated formally as follows.

**THEOREM 1 (Fusion algorithm)** [26] *Suppose  $\{\{\tau_1, \dots, \tau_m\}, \downarrow, \otimes\}$  is a VN where  $\tau_i$  is a valuation for  $t_i$ , and suppose  $\downarrow$  and  $\otimes$  satisfy Axioms A1–A3. Let  $\Psi$  denote  $t_1 \cup \cdots \cup t_m$ . Suppose  $t \subseteq \Psi$ , and suppose  $X_1 X_2 \cdots X_n$  is a sequence of variables in  $\Psi - t$ . Then*

$$(\tau_1 \otimes \cdots \otimes \tau_m)^{\downarrow t} = \otimes \left\{ \text{Fus}_{X_n} \left\{ \dots \text{Fus}_{X_2} \left\{ \text{Fus}_{X_1} \{\tau_1, \dots, \tau_m\} \right\} \right\} \right\}.$$

To illustrate Theorem 1, consider the VN of Example 1. Suppose we need to compute the marginal of the joint for  $\{T\}$ , i.e.,  $(\alpha \otimes o_A \otimes \sigma \otimes \tau \otimes \lambda \otimes \beta \otimes \varepsilon \otimes \xi \otimes \delta \otimes o_D)^{\downarrow \{T\}}$ . Consider the deletion sequence *XASDBLE*. First, after fusion with respect to  $X$ , we have  $\{\alpha, \tau, \sigma, \lambda, \beta, \varepsilon, \xi^{\downarrow \{E\}}, \delta, o_A, o_D\}$ . Second, after fusion with respect to  $A$ , we have  $\{(\alpha \otimes o_A \otimes \tau)^{\downarrow \{T\}}, \sigma, \lambda, \beta, \varepsilon, \xi^{\downarrow \{E\}}, \delta, o_D\}$ . Third, after fusion with respect to  $S$ , we have  $\{(\alpha \otimes o_A \otimes \tau)^{\downarrow \{T\}}, (\sigma \otimes \lambda \otimes \beta)^{\downarrow \{L, B\}}, \varepsilon, \xi^{\downarrow \{E\}}, \delta, o_D\}$ . Fourth, after fusion with respect to  $D$ , we have  $\{(\alpha \otimes o_A \otimes \tau)^{\downarrow \{T\}}, (\sigma \otimes \lambda \otimes$



$\beta)^{\downarrow_{\{L, B\}}}, \varepsilon, \xi^{\downarrow_{\{E\}}}, (\delta \otimes o_D)^{\downarrow_{\{E, B\}}}$ . Fifth, after fusion with respect to  $B$ , we have

$$\left\{ (\alpha \otimes o_A \otimes \tau)^{\downarrow_{\{T\}}}, \right. \\ \left. ((\sigma \otimes \lambda \otimes \beta)^{\downarrow_{\{L, B\}}} \otimes (\delta \otimes o_D)^{\downarrow_{\{E, B\}}})^{\downarrow_{\{L, E\}}}, \varepsilon, \xi^{\downarrow_{\{E\}}} \right\}.$$

Sixth, after fusion with respect to  $L$ , we have

$$\left\{ (\alpha \otimes o_A \otimes \tau)^{\downarrow_{\{T\}}}, \left( ((\sigma \otimes \lambda \otimes \beta)^{\downarrow_{\{L, B\}}} \otimes (\delta \otimes o_D)^{\downarrow_{\{E, B\}}})^{\downarrow_{\{L, E\}}} \right. \right. \\ \left. \left. \otimes \varepsilon \right)^{\downarrow_{\{T, E\}}}, \xi^{\downarrow_{\{E\}}} \right\}.$$

Finally, after fusion with respect to  $E$ , we have

$$\left\{ (\alpha \otimes o_A \otimes \tau)^{\downarrow_{\{T\}}}, \left[ \left( ((\sigma \otimes \lambda \otimes \beta)^{\downarrow_{\{L, B\}}} \otimes (\delta \otimes o_D)^{\downarrow_{\{E, B\}}})^{\downarrow_{\{L, E\}}} \otimes \varepsilon \right)^{\downarrow_{\{T, E\}}} \right. \right. \\ \left. \left. \otimes \xi^{\downarrow_{\{E\}}} \right] \right\}.$$

Theorem 4.1 tells us that

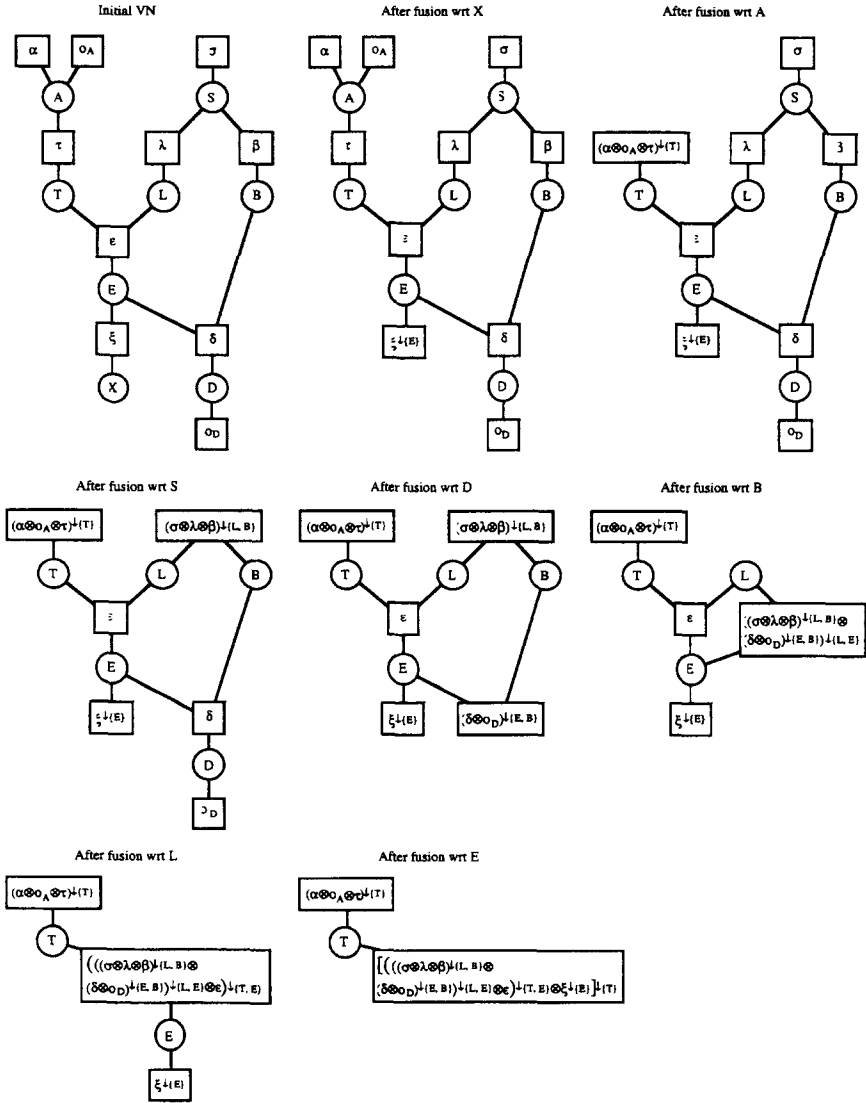
$$(\alpha \otimes o_A \otimes \tau)^{\downarrow_{\{T\}}} \otimes \left[ \left( ((\sigma \otimes \lambda \otimes \beta)^{\downarrow_{\{L, B\}}} \otimes (\delta \otimes o_D)^{\downarrow_{\{E, B\}}})^{\downarrow_{\{L, E\}}} \otimes \varepsilon \right)^{\downarrow_{\{T, E\}}} \right. \\ \left. \otimes \xi^{\downarrow_{\{E\}}} \right]^{\downarrow_{\{T\}}} \\ = (\alpha \otimes o_A \otimes \sigma \otimes \tau \otimes \lambda \otimes \beta \otimes \varepsilon \otimes \xi \otimes \delta \otimes o_D)^{\downarrow_{\{T\}}}.$$

The fusion algorithm is shown graphically in Figure 2.

### Deletion Sequences

Different deletion sequences may involve different computational efforts. Finding an optimal deletion sequence is a secondary optimization problem that has been shown to be NP-complete [2]. But there are several heuristics for finding good deletion sequences [11, 16, 34, 10].

One such heuristic is called one-step lookahead [17, 11]. This heuristic tells us which variable to delete next. According to this heuristic, the variable that should be deleted next is one that leads to combination over



**Figure 2.** The fusion algorithm for the VN of Example 1 using the deletion sequence *XASDBLE*.

the smallest state space, with ties broken arbitrarily. In particular, if a variable appears in the domain of only one valuation, then such variables should be deleted first, as no combination is involved.

For example, in the VN of Example 1, if we assume that each variable has a state space consisting of two configurations, then this heuristic would

pick  $X$  for the first deletion, since deletion of  $X$  involves no combination, whereas deleting any other variable involves some combination. The deletion sequence *XASDBLE* used to illustrate the fusion algorithm is one of the many deletion sequences suggested by the one-step lookahead heuristic.

---

## 5. FUSION ALGORITHM AS MESSAGE PASSING IN JOIN TREES

---

If we can compute the marginal of the joint valuation for one variable, then we can compute the marginals for all variables. We simply compute them one after the other. It is obvious, however, that this will involve much repetition of effort. To avoid this repetition, we will describe the fusion algorithm as message passing in join trees. A join tree can be thought of as a data structure that allows us to organize the computation, and more importantly, that allows us to cache the computations to avoid repetition of effort. In the next section, we will describe how the join-tree data structure allows us to compute multiple marginals efficiently.

### Join Trees

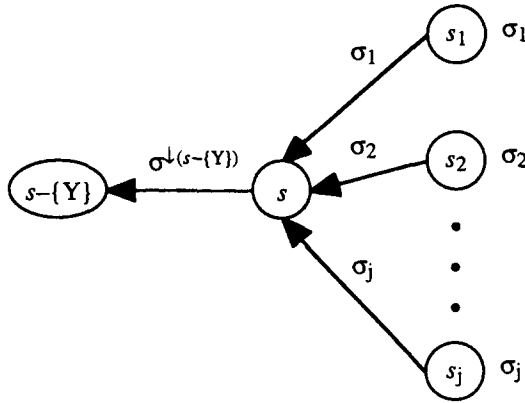
A *join tree* is a tree whose nodes are subsets of  $\Psi$  such that if a variable is in two distinct nodes, then it is in every node on the path between the two nodes [15]. As we will see, join trees are useful data structures to cache computation.

Consider again the definition of fusion. Suppose we have valuations  $\sigma_1, \dots, \sigma_k$ , where  $\sigma_i$  is a valuation for  $s_i$ . Suppose the valuations are labeled so that  $s_1, \dots, s_j$  contain  $Y$  and  $s_{j+1}, \dots, s_k$  do not contain  $Y$ . Then

$$\text{Fus}_Y\{\sigma_1, \dots, \sigma_k\} = \{\sigma^{\downarrow(s-\{Y\})}\} \cup \{\sigma_{j+1}, \dots, \sigma_k\},$$

where  $\sigma = \sigma_1 \otimes \dots \otimes \sigma_j$  and  $s = s_1 \cup \dots \cup s_j$ . We will now describe the fusion operation as message passing in a rooted join tree.

Suppose that  $s_1, \dots, s_k$ ,  $s$ , and  $s - \{Y\}$  are all distinct subsets of  $\Psi$ . We can describe the fusion operation as follows. We imagine that  $s_1, \dots, s_j$ ,  $s$ , and  $s - \{Y\}$  are all nodes connected together in a rooted join tree as shown in Figure 3, where the arrows point toward the root. Nodes  $s_1, \dots, s_j$  have stored in them the valuations  $\sigma_1, \dots, \sigma_j$ , respectively. Nodes  $s$  and  $s - \{Y\}$  have nothing stored in them. First nodes  $s_1, \dots, s_j$  send messages to their inward neighbor  $s$  consisting of the valuations stored in them. Next, node  $s$  first combines all messages it receives from its outward neighbors, marginalizes the combination  $\sigma$  to  $\sigma^{\downarrow(s-\{Y\})}$ , and sends this valuation as a message to its inward neighbor  $s - \{Y\}$ . At the end of the fusion operation, we are only concerned with node  $s - \{Y\}$  with a message  $\sigma^{\downarrow(s-\{Y\})}$ .



**Figure 3.** Fusion as message passing in a rooted join tree.

Is  $s - \{Y\}$  is not distinct from  $s_{j+1}, \dots, s_k$ , say  $s - \{Y\} = s_{j+1}$ , then the only change we need to make is that node  $s - \{Y\} = s_{j+1}$  will have valuation  $\sigma_{j+1}$  stored in it, but this valuation plays no role in fusion with respect to  $Y$ . Also, if not all  $s_1, \dots, s_j$  are distinct, say  $s_1 = s_2$ , then we have only one node for  $s_1$  and  $s_2$ , and we imagine that this node either has two valuations  $\sigma_1$  and  $\sigma_2$  stored in it or has one valuation  $\sigma_1 \otimes \sigma_2$  stored in it. The basic idea is that any subset of variables should appear only once in the rooted tree.<sup>1</sup>

In all cases, the rule for messages is stated as Rule 1.

**RULE 1 (Messages)** *Each node sends a message to its inward neighbor (toward the root). The message that a node sends to its inward neighbor is as follows. First it combines all messages it receives from its outward neighbors together with its own valuation (if any). Next it marginalizes the combination to the intersection of itself with its inward neighbor. Each node sends a message when it has received messages from all its outward neighbors. Leaves have no outward neighbors and can send messages right away.*

As we continue to delete variables using fusion, we recursively grow the rooted join tree. When we have deleted all but one variable, say  $X$ , we have a rooted join tree with  $\{X\}$  as the root. A formal description of the process of constructing a join tree is as follows.

<sup>1</sup>There is no problem with having a subset appear more than once in a rooted tree. But since there is no need for it at this stage, we try to avoid it. In Section 7, we will describe binary join trees that have multiple copies of subsets to make the message-passing algorithm more efficient.

### Rooted-Join-Tree Construction

Let  $\Psi$  denote the set of variables, let  $\Phi$  denote the set of subsets of variables for which we have valuations, let  $|\Phi|$  denote the number of elements of  $\Phi$ , let  $N$  denote the nodes of the rooted join tree, and let  $E$  denote the edges of the rooted join tree. A procedure in pseudocode for constructing a rooted join tree  $(N, E)$  for computing the marginal for  $\{X\}$  is as follows [25]:

PROCEDURE (Constructing a rooted join tree)

INPUT:  $\Psi, \Phi$

OUTPUT:  $N, E$

**INITIALIZATION**

$\Psi_u \leftarrow \Psi - \{X\}$  /\*  $\Psi_u$  denotes the set of variables in  $\Psi$  that have not yet been deleted \*/

$\Phi_u \leftarrow \Phi$  /\*  $\Phi_u$  denotes the subsets in  $\Phi$  that have not yet been arranged in the join tree \*/

$N \leftarrow \emptyset$

$E \leftarrow \emptyset$

**DO WHILE**  $|\Phi_u| > 1$  /\* If  $|\Phi_u| = 1$ , then we are done \*/  
     *Pick a variable  $Y \in \Psi_u$  /\* using some heuristic \*/*  
      $s \leftarrow \bigcup \{s_i \in \Phi_u \mid Y \in s_i\}$ .  
      $N \leftarrow N \cup \{s_i \in \Phi_u \mid Y \in s_i\} \cup \{s\} \cup \{s - \{Y\}\}$   
      $E \leftarrow E \cup \{(s_i, s) \mid s_i \in [\Phi_u - \{s\}], Y \in s_i\} \cup \{(s, s - \{Y\})\}$   
      $\Psi_u \leftarrow \Psi_u - \{Y\}$   
      $\Phi_u \leftarrow [\Phi_u - \{s_i \in \Phi_u \mid Y \in s_i\}] \cup \{s - \{Y\}\}$

**END DO**

$N \leftarrow N \cup \Phi_u$  /\* This is needed for the trivial case  $|\Phi| = 1$  \*/

**END**

The root node in the rooted join tree has no inward neighbor and does not send a message to any node. Instead, the root node simply combines all messages it receives from its outward neighbors, resulting in the desired marginal (as per Theorem 1). We describe this as Rule 2.

**RULE 2 (Marginal)** *When the root node has received a message from each of its outward neighbors, it combines all messages together with its own valuation and reports the result as its marginal.*

The construction of the join tree and Rules 1 and 2 completely describe the fusion algorithm in terms of message passing in join trees. Figure 4 illustrates the construction of the join tree and the messages for computing the marginal for  $T$  in the chest-clinic problem (Example 1).



singleton subset  $\{Z\}$ , then the same join tree (with some of the directions changed) will serve for the computation of the marginal for  $Z$ . We simply redirect some of the edges so that  $\{Z\}$  is now the root (instead of  $\{X\}$ ).

For example, for the valuation network of Example 1, if we would like the marginal of the joint for  $E$  instead of  $T$ , we have to reverse the directions of two edges— $((T, E), \{T\})$  and  $(\{E\}, \{T, E\})$  (see Figure 4). Suppose we have saved the messages computed for the marginal for  $\{T\}$ . Then, to compute the marginal for  $\{E\}$ , we have to compute only two new messages—the message from  $\{T\}$  to  $\{T, E\}$ , and the message from  $\{T, E\}$  to  $\{E\}$ . If we use Rule 1 to compute these messages, then the message from  $\{T\}$  to  $\{T, E\}$  is  $(\alpha \otimes o_A \otimes \tau)^{\downarrow_{(T)}}$ , and the message from  $\{T, E\}$  to  $\{E\}$  is

$$\left[ (\alpha \otimes o_A \otimes \tau)^{\downarrow_{(T)}} \otimes \left( ((\sigma \otimes \lambda \otimes \beta)^{\downarrow_{\{L, B\}}} \otimes (\delta \otimes o_D)^{\downarrow_{\{E, B\}}})^{\downarrow_{\{L, E\}}} \otimes \varepsilon \right)^{\downarrow_{\{T, E\}}} \right]^{\downarrow_E}.$$

If  $\{E\}$  now uses Rule 2, the marginal for  $\{E\}$  is given by

$$\left[ (\alpha \otimes o_A \otimes \tau)^{\downarrow_{(T)}} \otimes \left( ((\sigma \otimes \lambda \otimes \beta)^{\downarrow_{\{L, B\}}} \otimes (\delta \otimes o_D)^{\downarrow_{\{E, B\}}})^{\downarrow_{\{L, E\}}} \otimes \varepsilon \right)^{\downarrow_{\{T, E\}}} \right]^{\downarrow_E} \otimes \xi^{\downarrow_E}.$$

Instead of directing the edges of the join tree, it will be easier to leave the edges of the join tree undirected, and simply associate directions with the messages. Also, if each node sends a message to each of its neighbors, then we can compute the marginals for every subset in the join tree. We do this by changing the two rules as follows.

**RULE 1' (Messages)** *Each node sends a message to each of its neighbors. Suppose  $\mu^{r \rightarrow s}$  denotes the message from  $r$  to  $s$ , suppose  $N(r)$  denotes the neighbors of  $r$  in the join tree, and suppose the valuation associated with node  $r$  is denoted by  $\alpha_r$ . Then the message from node  $r$  to its neighboring node  $s$  is given as follows:*

$$\mu^{r \rightarrow s} = \left( \bigotimes \{ \mu^{t \rightarrow r} \mid t \in N(r) - \{s\} \} \otimes \alpha_r \right)^{\downarrow_{r \cap s}}. \quad (7.1)$$

*In words, the message that  $r$  send to its neighbor  $s$  is the combination of all messages that  $r$  receives from its other neighbors together with its own valuation suitably marginalized. Regarding timing, it is clear that node  $r$*

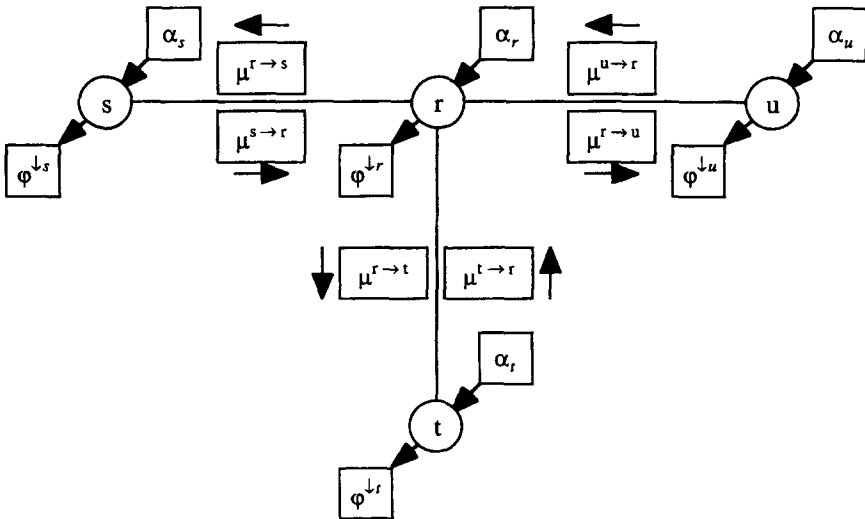
sends a message to neighbor  $s$  only when  $r$  has received a message from each of its other neighbors. A leaf of the join tree has only one neighbor, and therefore it can send a message to its neighbor right away without waiting for any messages.

**RULE 2' (Marginals)** When a node  $r$  has received a message from each of its neighbors, it combines all messages together with its own valuation and reports the results as its marginal. If  $\varphi$  denotes the joint valuation, then

$$\varphi^{\downarrow r} = \bigotimes \{\mu^{t \rightarrow r} | t \in N(r)\} \otimes \alpha_r \quad (7.2)$$

Using Rules 1' and 2', we can compute the marginal of the joint for each subset in the join tree. Thus, if we know the subsets for which we need marginals, we simply include these subsets (along with the subsets for which we have valuations) in  $\Phi$  during the construction of the join tree.

Rules 1' and 2' suggest an architecture as shown in Figure 5. Each node in the join tree would have two storage registers, one for the input valuation, and one for reporting the marginal of the joint. Also, each edge in the join tree would have two storage registers for the two messages, one in each direction.



**Figure 5.** An architecture for computing multiple marginals.



## 7. BINARY JOIN TREES

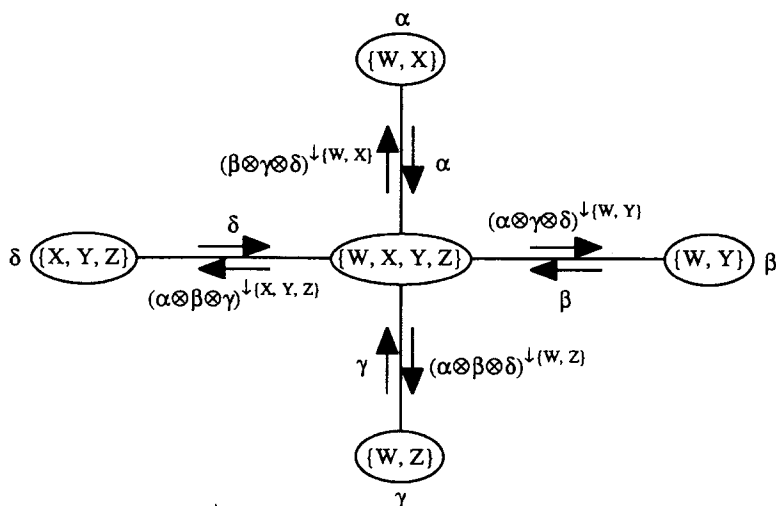
In this section, we introduce the concept of a binary join tree. Binary join trees are important from a computational viewpoint, since they reduce the number of computations involved in computing marginals.

A *binary join tree* is a join tree such that no node has more than three neighbors. The join tree shown in Figure 4 for the chest-clinic problem is not binary, since node  $\{S, L, B\}$  has four neighbors.

To explain the importance of a binary join tree, we will describe, by means of an example, the inefficiencies of computation in a nonbinary join tree.

**EXAMPLE 2** Consider a valuation network consisting of four variables  $W$ ,  $X$ ,  $Y$ , and  $Z$ , and four valuations:  $\alpha$  for  $\{W, X\}$ ,  $\beta$  for  $\{W, Y\}$ ,  $\gamma$  for  $\{W, Z\}$ , and  $\delta$  for  $\{X, Y, Z\}$ . A nonbinary join tree with the messages between adjacent nodes is shown in Figure 6. We make some observations about inefficiencies of computation in this nonbinary join tree.

1. *Domain of combination.* First, consider the message  $(\alpha \otimes \beta \otimes \gamma)^{\downarrow\{X, Y, Z\}}$  (from  $\{W, X, Y, Z\}$  to  $\{X, Y, Z\}$ ). The computation of this message involves combination of the valuations  $\alpha$ ,  $\beta$ , and  $\gamma$  on the domain  $\{W, X, Y, Z\}$ . In general, combination of  $m$  valuations on a domain with  $n$  configurations involves computation that is linear in



**Figure 6.** A nonbinary join tree with messages for the VN in Example 2.

$m - 1$  and a monotonic increasing function of  $n$ . Suppose that  $W$  has 2 states,  $X$  has 3 states, and  $Y$  has 4 states, and  $Z$  has 5 states. Then the state space of  $\{W, X, Y, Z\}$  has 120 configurations. Instead of combining  $\alpha$ ,  $\beta$ , and  $\gamma$  on the domain  $\{W, X, Y, Z\}$  that has 120 configurations, it is more efficient to first combine  $\alpha$  and  $\beta$  on domain  $\{W, X, Y\}$  with 24 configurations, and next combine  $\alpha \otimes \beta$  with  $\gamma$  on the domain  $\{W, X, Y, Z\}$  with 120 configurations. A similar observation can be made for the message  $(\alpha \otimes \beta \otimes \delta)^{\downarrow_{\{W, Z\}}}$ .

2. *Nonlocal combination.* Second, consider the message  $(\beta \otimes \gamma \otimes \delta)^{\downarrow_{\{W, X\}}}$ . Notice that  $Z$  is in the domain of  $\gamma$  and  $\delta$ , but not in the domain of  $\beta$ . Thus it follows from Axiom A3 that

$$(\beta \otimes \gamma \otimes \delta)^{\downarrow_{\{W, X\}}} = (\beta \otimes (\gamma \otimes \delta)^{\downarrow_{\{W, X, Y\}}})^{\downarrow_{\{W, X\}}}.$$

It is computationally more efficient to compute  $(\beta \otimes (\gamma \otimes \delta)^{\downarrow_{\{W, X, Y\}}})^{\downarrow_{\{W, X\}}}$  than to compute  $(\beta \otimes \gamma \otimes \delta)^{\downarrow_{\{W, X\}}}$ . Similarly, instead of computing  $(\alpha \otimes \gamma \otimes \delta)^{\downarrow_{\{W, Y\}}}$ , it is more efficient to compute instead  $(\alpha \otimes (\gamma \otimes \delta)^{\downarrow_{\{W, X, Y\}}})^{\downarrow_{\{W, Y\}}}$ .

3. *Repetition of combinations.* Third, consider the messages  $(\alpha \otimes \beta \otimes \gamma)^{\downarrow_{\{X, Y, Z\}}}$  and  $(\alpha \otimes \beta \otimes \delta)^{\downarrow_{\{W, Z\}}}$ . Notice that if these two messages are computed separately, then the combination of  $\alpha$  and  $\beta$  is repeated. Also, for messages  $(\beta \otimes \gamma \otimes \delta)^{\downarrow_{\{W, X\}}}$  and  $(\alpha \otimes \gamma \otimes \delta)^{\downarrow_{\{W, Y\}}}$  the combination of  $\gamma$  and  $\delta$  is repeated [32, 33].

Now consider a binary join tree for the same VN as shown in Figure 7. Compared to the nonbinary join tree of Figure 6, the binary join tree has an additional node  $\{W, X, Y\}$  and an additional edge  $(\{W, X, Y\}, \{W, X, Y, Z\})$ .

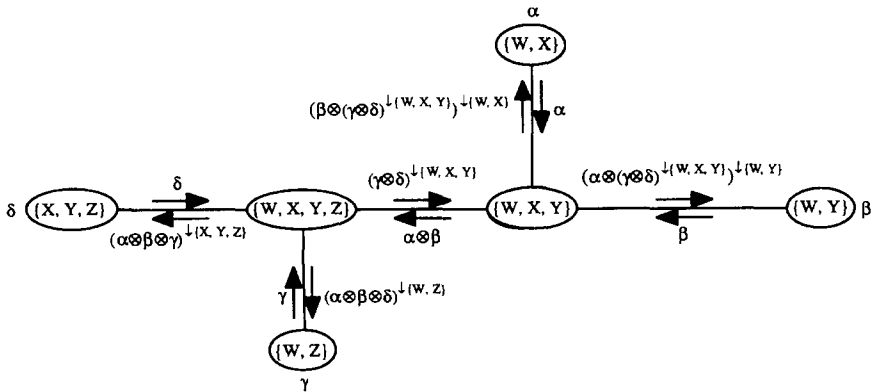


Figure 7. A binary join tree with messages for the VN in Example 2.

First, notice that  $\alpha \otimes \beta$  is computed on the domain  $\{W, X, Y\}$  (as a message from  $\{W, X, Y\}$  to  $\{W, X, Y, Z\}$ ) before we compute  $(\alpha \otimes \beta \otimes \gamma)^{\downarrow_{\{X, Y, Z\}}}$  (as a message from  $\{W, X, Y, Z\}$  to  $\{X, Y, Z\}$ ) and  $(\alpha \otimes \beta \otimes \delta)^{\downarrow_{\{W, Z\}}}$  (as a message from  $\{W, X, Y, Z\}$  to  $\{W, Z\}$ ). Thus we avoid combining valuations on domains bigger than is necessary.

Second, instead of computing  $(\beta \otimes \gamma \otimes \delta)^{\downarrow_{\{W, X\}}}$ , we compute  $(\beta \otimes (\gamma \otimes \delta))^{\downarrow_{\{W, X, Y\}} \downarrow_{\{W, X\}}}$ , and instead of computing  $(\alpha \otimes \gamma \otimes \delta)^{\downarrow_{\{W, Y\}}}$ , we compute  $(\alpha \otimes (\gamma \otimes \delta))^{\downarrow_{\{W, X, Y\}} \downarrow_{\{W, Y\}}}$ . Thus the messages are computed locally.

Third, the combination  $(\gamma \otimes \delta)^{\downarrow_{\{W, X, Y\}}}$  that appears in the messages  $(\beta \otimes (\gamma \otimes \delta))^{\downarrow_{\{W, X, Y\}} \downarrow_{\{W, X\}}}$  and  $(\alpha \otimes (\gamma \otimes \delta))^{\downarrow_{\{W, X, Y\}} \downarrow_{\{W, Y\}}}$  is computed only once. Also, the combination  $\alpha \otimes \beta$  is computed only once for the messages  $(\alpha \otimes \beta \otimes \gamma)^{\downarrow_{\{X, Y, Z\}}}$  and  $(\alpha \otimes \beta \otimes \delta)^{\downarrow_{\{W, Z\}}}$ . Thus we avoid repetition of combinations.

For these three reasons, binary join trees are a more efficient way to organize the computations than nonbinary join trees.

How does one construct a binary join tree? We will describe a technique based on the idea of binary fusion.

### Binary Fusion

Consider again the definition of fusion. Suppose we have valuations  $\sigma_1, \dots, \sigma_k$ , where  $\sigma_i$  is a valuation for  $s_i$ . Suppose the valuations are labeled so that  $s_1, \dots, s_j$  contain  $Y$  and  $s_{j+1}, \dots, s_k$  do not contain  $Y$ . Then

$$\text{Fus}_Y\{\sigma_1, \dots, \sigma_k\} = \{\sigma^{\downarrow_{(s - \{Y\})}}\} \cup \{\sigma_{j+1}, \dots, \sigma_k\}$$

where  $\sigma = \sigma_1 \otimes \dots \otimes \sigma_j$ , and  $s = s_1 \cup \dots \cup s_j$ . In binary fusion, we compute  $\sigma$  by combining valuations two at a time recursively as follows. We start with the set of valuations we need to combine to compute  $\sigma$ , namely  $\{\sigma_1, \dots, \sigma_j\}$ . Let  $\|s_i\|$  denote the number of configurations in the state space of  $s_i$ . Suppose that the subsets are labeled so that  $\|s_1 \cup s_2\| = \text{argmin}\{\|s_p \cup s_q\| \mid 1 \leq p, q \leq j\}$ . Then we first combine  $\sigma_1 \otimes \sigma_2$ . Now the set of valuations that need to be combined is  $\{\sigma_1 \otimes \sigma_2, \sigma_3, \dots, \sigma_j\}$ . We repeat this procedure recursively till we end with a set of one valuation,  $\{\sigma\}$ . Binary fusion is essentially fusion with the added requirement that all combinations be done on a binary basis, i.e., two at a time.

As in the case of fusion, we can implement binary fusion as a message passing scheme in join trees. The join tree that is suggested by binary fusion is, of course, binary. For example, consider the chest-clinic problem in Example 1. After fusion with respect to  $X$  and  $A$ , we have valuations  $\{(\alpha \otimes o_A \otimes \tau)^{\downarrow_{\{T\}}}, \sigma, \lambda, \beta, \varepsilon, \xi^{\downarrow_{\{E\}}}, \delta, o_D\}$ . Next we wish to do fusion with respect to  $S$ . Ordinary fusion with respect to  $S$  leads to a nonbinary join

tree (see Figure 4). If we do binary fusion with respect to  $S$ , then to combine  $\sigma$ ,  $\lambda$ , and  $\beta$ , we first combine  $\sigma$  and  $\lambda$ , and then we combine  $\sigma \otimes \lambda$  and  $\beta$ . We can represent this as a message-passing scheme in a binary join tree as shown in Figure 8.

In the above procedure, notice that some subsets may appear more than once in the join tree. For example,  $s_1 \cup s_2$  may be identical with, say,  $s_2$ . This poses no problem in practice. However, to accommodate this, we need to consider the set of nodes of a join tree as a multiset, i.e., a set of indexed subsets of  $\Psi$  (two identical subsets corresponding to distinct nodes will be identified unambiguously with their indices, which will be different).

### Binary-Join-Tree Construction

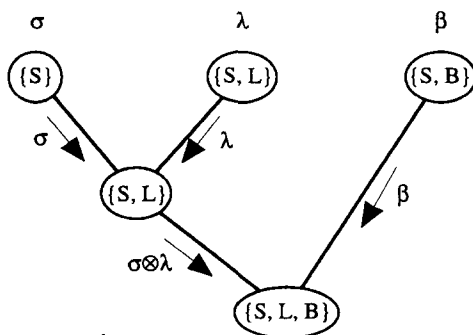
We will now formally describe a procedure in pseudocode that describes a construction of the binary join tree suggested by binary fusion.

Let  $\Psi$  denote the set of variables, let  $\Phi$  denote the multiset of subsets of variables for which we have valuations or the subsets for which we need marginals, let  $N$  denote the multiset of nodes of the binary join tree and  $E$  denote the edges of the binary join tree, let  $|\Phi|$  denote the number of elements of the set  $\Phi$ , and let  $\|s\|$  denote the number of elements of the state space of the subset  $s$ . We assume the subsets in  $\Phi$  are indexed  $1, \dots, k$ , where  $k = |\Phi|$ , i.e.,  $\Phi = \{s_1, \dots, s_k\}$ . A procedure in pseudocode for constructing a binary join tree  $(N, E)$  using binary fusion is as follows.

PROCEDURE (Constructing a binary join tree)

INPUT:  $\Psi, \Phi$

OUTPUT:  $N, E$



**Figure 8.** Binary fusion as message passing in a binary join tree.

**INITIALIZATION**

$\Psi_u \leftarrow \Psi$  /\*  $\Psi_u$  denotes the set of variables in  $\Psi$  that have not yet been deleted \*/

$\Phi_u \leftarrow \Phi$  /\*  $\Phi_u$  denotes the multiset of subsets, indexed as  $s_1, \dots, s_{|\Phi|}$ , that have yet to be arranged in the join tree \*/

$N \leftarrow \emptyset$  /\*  $N$  denotes the multiset of nodes of the binary join tree \*/

$E \leftarrow \emptyset$  /\*  $E$  denotes the set of edges \*/

$k \leftarrow |\Phi| + 1$  /\*  $k$  will be the index of the next subset created in the process of constructing a binary join tree \*/

**DO WHILE**  $|\Phi_u| > 1$  /\* If  $|\Phi_u| = 1$ , then we are done \*/

Pick a variable  $Y \in \Psi_u$  (using some heuristic)

$\Phi_Y \leftarrow \{s_i \in \Phi_u \mid Y \in s_i\}$ .

**DO WHILE**  $|\Phi_Y| > 1$

$r_1 \leftarrow s_i$  and  $r_2 \leftarrow s_j$  where  $s_i, s_j \in \Phi_Y$  and  $\|s_i \cup s_j\| \leq \|s_p \cup s_q\|$   
for all  $s_p, s_q \in \Phi_Y$

$s_k \leftarrow r_1 \cup r_2$  /\*  $s_k$  is the new indexed subset created \*/

$N \leftarrow N \cup \{r_1, r_2, s_k\}$

$E \leftarrow E \cup \{\{r_1, s_k\}, \{r_2, s_k\}\}$

$\Phi_Y \leftarrow \Phi_Y - \{r_1, r_2\} \cup \{s_k\}$  /\* Each time the number of subsets in  $\Phi_Y$  is reduced by exactly 1 \*/

$k \leftarrow k + 1$

**END DO**

**IF**  $|\Psi_u| > 1$  **THEN DO**

$r \leftarrow s_i$  where  $\{s_i\} = \Phi_Y$

$s_k \leftarrow r - \{Y\}$  /\*  $s_k$  is the new indexed subset created \*/

$k \leftarrow k + 1$

$N \leftarrow N \cup \{r\} \cup \{s_k\}$

$E \leftarrow E \cup \{\{r, s_k\}\}$

$\Phi_u \leftarrow \Phi_u \cup \{s_k\} - \{s_i \in \Phi_u \mid Y \in s_i\}$

$\Psi_u \leftarrow \Psi_u - \{Y\}$

**END IF**

**END DO**

$N \leftarrow N \cup \Phi_u$  /\* This is needed for the trivial case  $|\Phi| = 1$  \*/

**END**

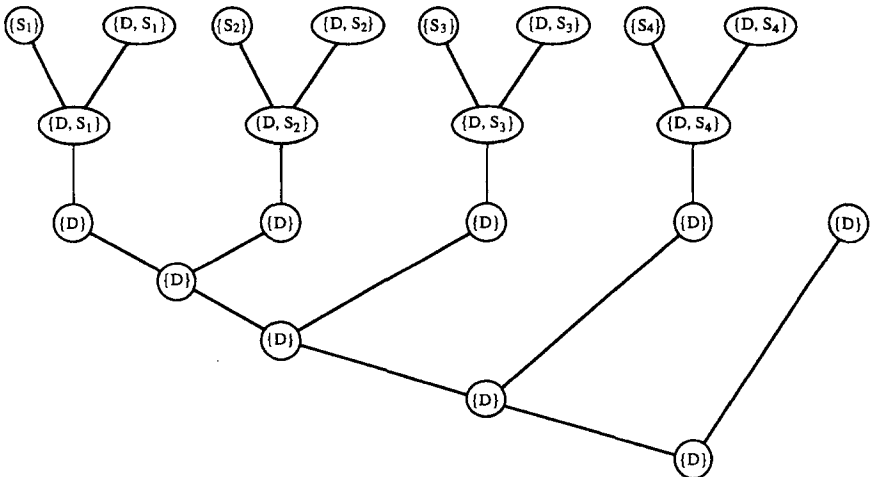
It is easy to see from the procedure that the join tree constructed will always be binary. We illustrate this procedure with an example.

**EXAMPLE 3** Consider a valuation network with valuations as follows:  $\delta$  for  $\{D\}$ ,  $\sigma_1$  for  $\{D, S_1\}$ ,  $\sigma_2$  for  $\{D, S_2\}$ ,  $\sigma_3$  for  $\{D, S_3\}$ ,  $\sigma_4$  for  $\{D, S_4\}$ ,  $\sigma_1$  for  $\{S_1\}$ , and  $\sigma_2$  for  $\{S_2\}$ . Suppose we need the marginal of the joint for all five variables. If we implement the binary fusion procedure for the multiset  $\Phi = \{\{D\}, \{D, S_1\}, \{D, S_2\}, \{D, S_3\}, \{D, S_4\}, \{S_1\}, \{S_2\}, \{S_4\}, \{S_4\}\}$  using deletion sequence  $S_1 S_2 S_3 S_4 D$  suggested by the one-step lookahead heuristic, the resulting join tree is displayed in Figure 9.

## 8. CONCLUSIONS

The main goal of this paper is to describe a data structure called binary join trees that are useful in computing multiple marginals efficiently using the Shenoy-Shafer architecture. We define binary join trees, describe their utility, and describe a procedure for constructing them.

The join-tree construction process described here is superficially different from the method described in Lauritzen and Spiegelhalter [12], which consists of moralizing a directed acyclic graph, triangulating the moral graph using the maximum-cardinality search method, and then arranging the cliques of the triangulated moral graph in a join tree. Instead of starting from a directed acyclic graph, we start with a more general setting—a hypergraph consisting of all subsets for which we have valuations and



**Figure 9.** A binary join tree resulting from the binary fusion procedure applied to the valuation network in Example 3.

all subsets for which we desire marginals, and we use the fusion algorithm as a guide to constructing a join tree. Thus the join tree can be regarded as a data structure to organize the computations of the fusion algorithm.

Binary join trees further refine the data structure so that unnecessary computations are minimized. In particular, we identify three sources of inefficiencies associated with nonbinary join trees that are eliminated in binary join trees, and we describe an automatic procedure for constructing them based on the idea of binary fusion.

Although we have restricted our discussion in this paper to the Shenoy-Shafer architecture, binary join trees are also useful in the Lauritzen-Spiegelhalter [12] and Hugin [9, 13] architectures, albeit to a lesser extent. This is because in the Lauritzen-Spiegelhalter and Hugin architectures, some of the combination operations are replaced by division operations. Since the main role of the binary join trees is to reduce the computation in combinations, their utility is less in these other architectures.

An interesting question is the relative computational efficiencies of the three architectures. It is commonly believed that the Hugin architecture is the most efficient (at least for the case of probabilistic reasoning). Ongoing work [14] leads us to believe that if binary join trees are used, then the Shenoy-Shafer architecture is at least as computationally efficient as—if not more than—the Hugin architecture.

---

## ACKNOWLEDGMENTS

---

This work is based on research supported in part by Hughes Research Laboratories under grant GP3044.95-125. Any opinions, findings, and conclusion or recommendations expressed in this paper are those of the author and do not necessarily reflect the views of Hughes Research Laboratories. I am grateful to Liping Liu, Hong Xu, Steffen L. Lauritzen, Finn V. Jensen, Peter Gillett, Kurt Reiser, and Yang Chen for comments and discussions.

---

## References

---

1. R. G. Almond, *Graphical Belief Modeling*, Chapman & Hall, London, 1995.
2. S. Arnborg, Corneil, D. G., and Proskurowski, A., Complexity of finding embeddings in a  $k$ -tree, *SIAM J. Algebraic Discrete Methods* 8, 277–284, 1987.
3. C. Beeri, Fagin, R., Maier, D., and Yannakakis, M., On the desirability of acyclic database schemes, *J. Assoc. Comput. Mach.* 30(3), 479–513, 1983.
4. U. Bertele, and Brioschi, F., *Nonserial Dynamic Programming*, Academic, New York, 1972.

5. C. Cannings, Thompson, E. A., and Skolnick, M. H., Probability functions on complex pedigrees, *Adv. Appl. Probab.* 10, 26–61, 1978.
6. J. Cano, Delgado, M., and Moral, S., An axiomatic framework for propagating uncertainty in directed acyclic networks, *Internat. J. Approx. Reason.* 8(4), 253–280, 1983.
7. A. P. Dempster, Upper and lower probabilities induced by a multivalued mapping, *Ann. Math. Statist.* 38, 325–339, 1967.
8. A. P. Dempster, and Kong, A., Uncertain evidence and artificial analysis, *J. Statist. Plann. Inference* 20, 355–368, 1988.
9. F. V. Jensen, Lauritzen, S. L., and Olesen, K. G., Bayesian updating in causal probabilistic networks by local computation, *Comput. Statist. Quart.* 4, 269–282, 1990.
10. U. Kjærulff, Triangulation of graphs—algorithms giving small total state space, Tech. Report R90-09, Inst. for Electronic Systems, Aalborg Univ., Denmark.
11. A. Kong, Multivariate belief functions and graphical models, PhD Dissertation, Dept. of Statistics, Harvard Univ., Cambridge, Mass., 1986.
12. S. L. Lauritzen, and Spiegelhalter, D. J., Local computations with probabilities on graphical structures and their application to expert systems (with discussion), *J. Roy. Statist. Soc. Ser. B* 50(2), 157–224, 1988.
13. S. L. Lauritzen, and Jensen, F. V., Local computation with valuations from a commutative semigroup, Tech. Report R-96-2028, Dept. of Mathematics and Computer Science, Aalborg Univ., Denmark.
14. V. Lepar, and Shenoy, P. P., A comparison of architectures for exact computation of marginals using local computation, Working Paper, School of Business, Univ. of Kansas, Lawrence, 1996.
15. D. Maier, *The Theory of Relational Databases*, Computer Science Press, 1983.
16. K. Mellouli, On the propagation of beliefs in networks using the Dempster-Shafer theory of evidence, PhD Dissertation, School of Business, Univ. of Kansas, Lawrence, 1987.
17. S. M. Olmsted, On representing and solving decision problems, PhD Thesis, Dept. of Engineering-Economic Systems, Stanford Univ., Stanford, Calif., 1983.
18. J. Pearl, Fusion, propagation and structuring in belief networks, *Artificial Intelligence* 29, 241–288, 1986.
19. D. J. Rose, Symmetric elimination on sparse positive definite systems and the potential flow network problem, PhD Dissertation, Harvard Univ., 1970.
20. G. Shafer, An axiomatic study of computation in hypertrees, Working Paper 232, School of Business, Univ. of Kansas, Lawrence, 1991.
21. G. Shafer, *Probabilistic Expert Systems*, SIAM, Philadelphia, 1996.



22. G. Shafer, Shenoy, P. P., and Mellouli, K., Propagating belief functions in qualitative Markov trees, *Internat. J. Approx. Reason.* 1(4), 349–400, 1987.
23. P. P. Shenoy, A valuation-based language for expert systems, *Internat. J. Approx. Reason* 3(5), 383–411, 1989.
24. P. P. Shenoy, On Spohn's rule for revision of beliefs, *Internat. J. Approx. Reason* 5(2), 149–181, 1991.
25. P. P. Shenoy, Valuation-based systems for discrete optimization, in *Uncertainty in Artificial Intelligence* 6 (P. P. Bonissone, M. Henrion, L. N. Kanal, and J. F. Lemmer, Eds.), North-Holland, Amsterdam, 385–400, 1991.
26. P. P. Shenoy, Valuation-based systems: A framework for managing uncertainty in expert systems, in *Fuzzy Logic for the Management of Uncertainty* (L. A. Zadeh, and J. Kacprzyk, Eds.), Wiley, New York, 83–104, 1992.
27. P. P. Shenoy, Using possibility theory in expert systems, *Fuzzy Sets and Systems* 52(2), 129–142, 1992.
28. P. P. Shenoy, Conditional independence in valuation-based systems, *Internat. J. Approx. Reason.* 10(3), 203–234, 1994.
29. P. P. Shenoy, Representing conditional independence relations by valuation networks, *Internat. J. Uncertain. Fuzziness Knowledge-Based Systems*, 2(2), 143–165, 1994.
30. P. P. Shenoy, Consistency in valuation-based systems, *ORSA J. Comput.* 6(3), 281–291, 1994.
31. P. P. Shenoy, and Shafer, G., Axioms for probability and belief-function propagation, in *Uncertainty in Artificial Intelligence* 4 (R. D. Shachter, T. S. Levitt, J. F. Lemmer, and L. N. Kanal, Eds.), North-Holland, Amsterdam, 169–198, 1990; reprinted in *Readings in Uncertain Reasoning* (G. Shafer and J. Pearl, Eds.), Morgan Kaufmann, San Mateo, Calif., 575–610, 1990.
32. H. Xu, An efficient implementation of belief function propagation, *Uncertainty in Artificial Intelligence: Proceedings of the Seventh Conference* (B. D'Ambrosio, P. Smets, and P. P. Bonissone, Eds.), Morgan Kaufmann, San Mateo, Calif., 425–432, 1991.
33. H. Xu, and Kennes, R., Steps toward efficient implementation of Dempster-Shafer theory, in *Advances in the Dempster-Shafer Theory of Evidence* (R. R. Yager, J. Kacprzyk, and M. Fedrizzi, Eds.), Wiley, New York, 153–174, 1994.
34. L. Zhang, Studies on finding hypertree covers of hypergraphs, Working Paper 198, School of Business, Univ. of Kansas, Lawrence, 1988.